As duas linguagens permitem modelagem em nível de portas lógicas, contudo VERILOG é mais indicada que VHDL para uma descrição Low-Level. Possuindo primitivas nativas ou portas lógicas, nas quais um desenvolvedor pode instanciar sem maiores traumas, quando comparada com VHDL. A dica para modelar Low-Level em VHDL é utilizar os operadores lógicos NOT, AND, NAND, OR, NOR, XOR, XNOR. suporte a duas dúzias de primitivas para modelagem de lógica estrutural. Adicionalmente a essas primitivas, e também suporte a primitivas definidas por usuários, UDP (*User-Defined Primitives)*. UDPs são muito úteis e populares entre projetistas de ASICs.

Em contrapartida,VHDL é muito melhor para descrever *hardware* em alto-nível,possibilita **a** criação de tipo de dados definidos pelo usuário**;** Pacotes (package) para reuso em projetos**:** Subprogramas e tipos de dados reutilizados podem ser declarados em um pacote VHDL e reutilizados em outras entidades e arquiteturas. VERILOG possui o equivalente através da diretiva de compilação ***include " "***; **Declaração de configuração em VHDL,** permite que um projeto em VHDL possua muitas entidades de projetos com arquiteturas diferentes. Este último recurso é muito útil para os projetistas que utilizam VHDL e precisam gerenciar um projeto de alto nível, cujas características podem variar; E por fim **Gerenciamento de Bibliotecas**:Essas possuem arquiteturas, entidades, pacotes e configurações já compiladas. Esse recurso é muito útil no gerenciamento de grandes estruturas de projeto.

VHDL é fortemente tipada, ou seja, você tem que descrever a sua lógica de acordo com o tipo de dados definidos na estrutura inicial do seu código. Caso você misture tipos de dados, o compilador irá acusar erro. Isso não ocorre em VERILOG - portanto muito cuidado e atenção na codificação em VERILOG. Outro aspecto que traduz em um VHDL mais "*verboso*" é o fato de o VHDL possuir muitos tipos de dados complexos, e, além disso, temos que considerar que o usuário pode usar outros tantos tipos de dados.

Pontualmente, ainda podemos listar que:

- VERILOG é muito parecido com a linguagem C, enquanto VHDL é muito próximo a ADA e ao PASCAL;
- VERILOG é *case-sensitive*, o que não ocorre com VHDL;
- Em VERILOG para usar um componente em um módulo basta fazer uma instância e fazer correto "*port_map"*. Em VHDL, para instanciar um componente é necessário declarar as bibliotecas, os pacotes, a entidade, seguido da arquitetura, para finalmente realizar o port-map dos sinais;
- VERILOG possui diversas diretivas de compilação como "*timescale"*, *"ifdef"*, "*else"*, "*include"*. Diretivas de compilação não foram permitidas em VHDL até a especificação VHDL2008. Aqui vale uma observação lembrada pelo engenheiro Ricardo Fialho - "*...sempre foi possível colocar código dentro de uma architecture usando procedure e functions...mas só servem para alguns casos...*".

  https://www.embarcados.com.br/verilog-vs-vhdl/